Jonathan Skaggs

CS 478 Perceptron

Arff Files

Linearly separable data:

```
% 1. Title: Testing Not Linearly Separable Database

@RELATION ls

@ATTRIBUTE x   Continuous
@ATTRIBUTE y   Continuous
@ATTRIBUTE class   {class1,class2}

@DATA
-0.8,  0.2, class1
 0.8, -0.2, class2
-0.5, -0.4, class1
 0.6,  0.3, class2
 0.1, -0.2, class1
 0.2, -0.1, class2
-1.0, -0.1, class1
 1.0,  0.8, class2
%
%
%
```

Not linearly separable data:

```
% 1. Title: Testing Linearly Separable Database

@RELATION  nls

@ATTRIBUTE x   Continuous
@ATTRIBUTE y   Continuous
@ATTRIBUTE class   {class1,class2}

@DATA
 0.3,  0.2, class1
-0.4,  0.5, class2
-0.6, -0.2, class1
 0.7, -0.4, class2
-0.3, -0.3, class1
 0.9, -0.7, class2
 0.4,  0.6, class1
-0.4,  0.7, class2
%
%
%
```

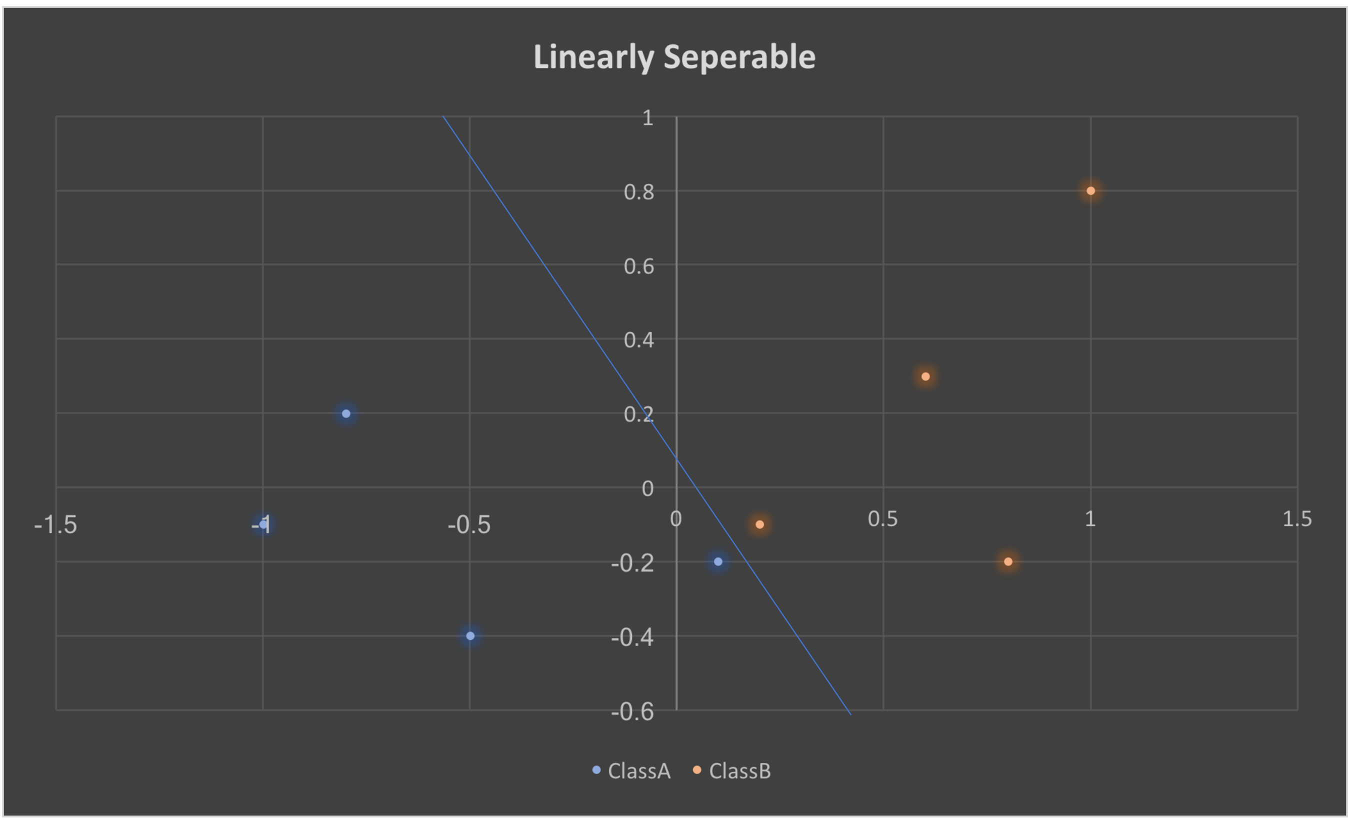
Stopping Criteria

I stopped when the sum of the log of the absolute value of each weight (sum(log(abs(weights)))) changed by less than a factor of .0001 over 5 iterations. I found that it was effective at stopping once it had converged but did not stop to early. (With different learning rates I needed to use a different change factor).

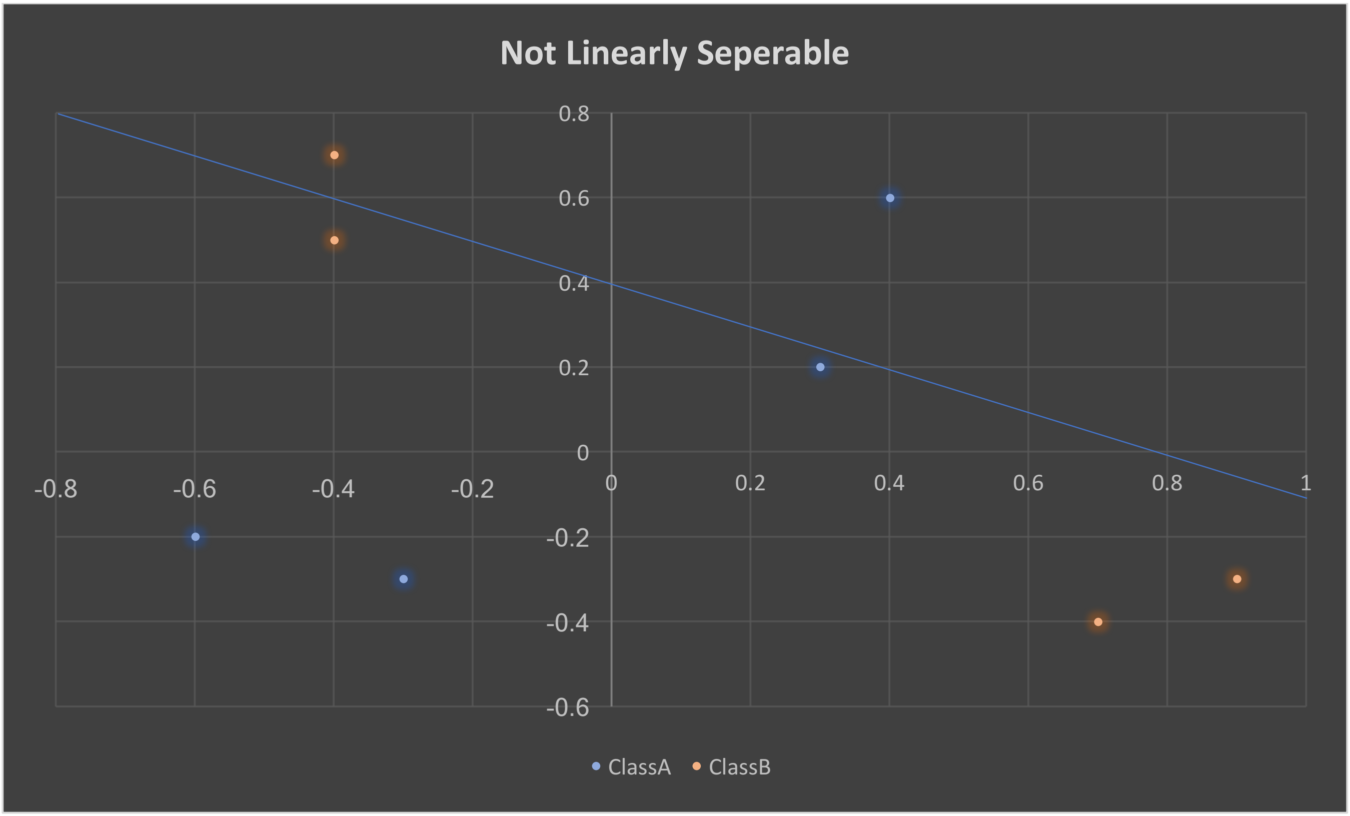
Learning Rates

As the training rate goes down the time taken increases as does the number of epochs. With c equal to 0.1 it would run consistently in about 20 epochs (there is a random aspect). When c was 0.01 and 0.001 it would take about 60 and 200 epochs respectively. Note: when the learning rate was so low I had to alter the stopping criteria because it began to stop prematurely. With c equal to 1 it would occasionally overcorrect and jump back and forth and not always converge correctly. (My linearly separable points are close together, see the figure above).

Graphs



0.526162842463x 1.14647270173y = 0.106005948126



0.0543648335254x + 0.122098663595y = 0.0415516890012

Voting Dataset

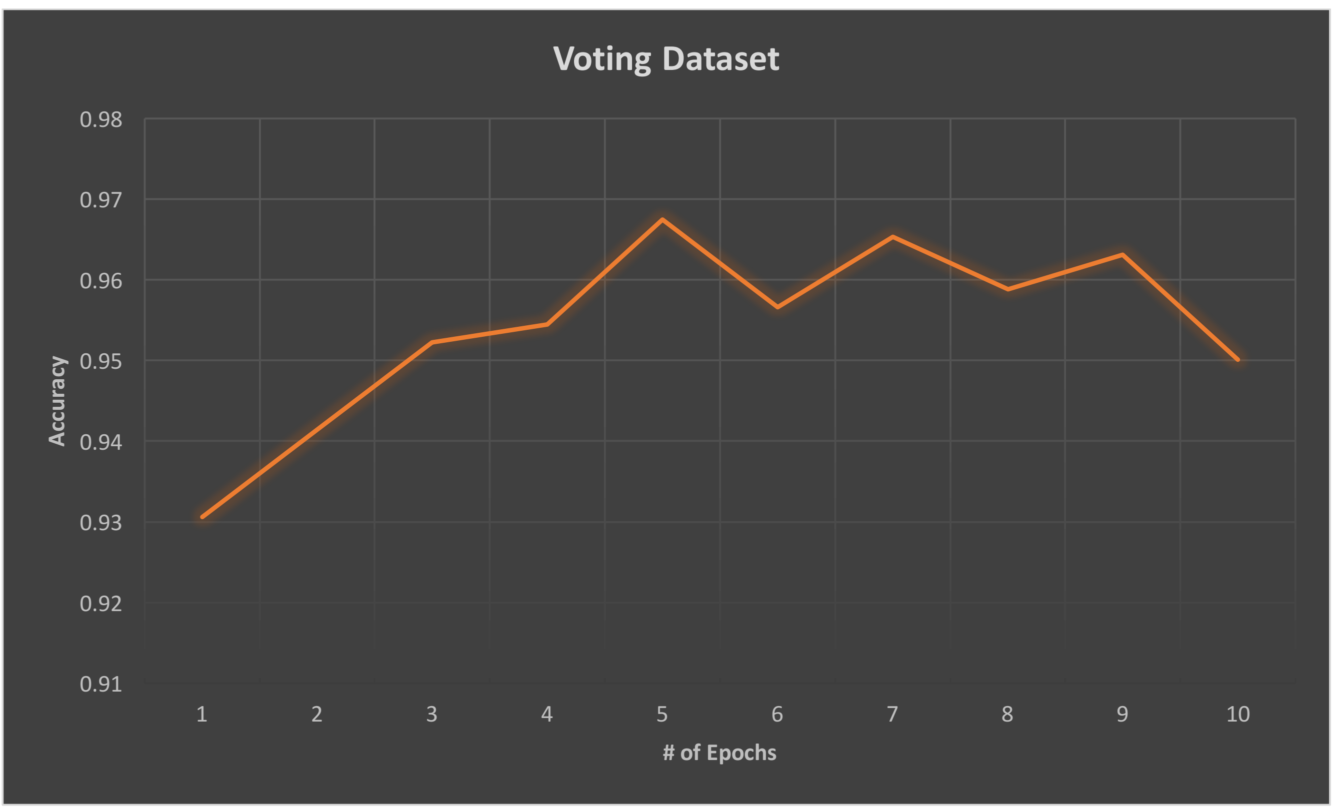
Percentage for training	0.7	0.7	0.7	0.7	0.7
Percentage for testing	0.3	0.3	0.3	0.3	0.3
Epochs	11	11	21	21	3
Time to train (in seconds)	0.959627329	0.017128944	0.0332129	0.035296917	0.002919912
Training set accuracy	0.942446043	0.953416149	0.972049689	0.95652173913	0.931677019
Test set accuracy	0.920863309	0.971223022	0.913669065	0.964028777	0.942446043

Average Testing Results

Percentage used for training	0.7
Percentage used for testing	0.3
Epochs	13
Time to train (in seconds)	0.209637201
Training set accuracy	0.949897225
Test set accuracy	0.942446043
handicapped-infants	-0.132931299
water-project-cost-sharing	-0.440079435

adoption-of-the-budget-resolution	-0.865280005
physician-fee-freeze	1.919873642
el-salvador-aid	0.61890204
religious-groups-in-schools	-0.228061737
anti-satellite-test-ban	0.565439888
aid-to-nicaraguan-contras	0.997197118
mx-missile	-0.859531055
immigration	0.45659763
synfuels-corporation-cutback	-1.096486767
education-spending	0.047081398
superfund-right-to-sue	0.096323497
crime	-0.070773242
duty-free-exports	-0.649793822
export-administration-act-south-africa	0.389240253

How people voted on "physician-fee-freeze" was the most telling of whether they would vote Republican or Democrat. You can tell because its weight has the largest absolute value. Education-spending, superfund-right-to-sue, and crime were the least critical features. You can tell this because the average weights are the closest to zero.



Iris Dataset

I created 1 perceptron for each pair of output classes, where the training set only contains examples from the 2 classes. I ran all perceptrons on novel data and set the class to the label with the most wins (votes) from the perceptrons. In case of a tie, I used the net values to decide.

Dataset name: ../datasets/iris.arff
Number of instances: 150
Number of attributes: 5
Calculating accuracy on training set...
Time to train (in seconds): 11.1815319061
Training set accuracy: 0.966666666667